Virtual Functions and Polymorphism Exercises

Copyright © James Raynard 2021

Static and dynamic types

- Explain what is meant by static and dynamic binding when calling a member function on an object
- What two conditions must apply for dynamic binding to occur?

Copyright © James Raynard 2021

Virtual Member Functions

- What is a virtual member function?

Copyright © James Raynard 2021

Function call arguments

- Using the code on the next page, write a program which creates a Circle object and passes it to draw_shape()
- Make the draw() member function in Drawable virtual and run your program again
- Add a Triangle class, which is also derived from Drawable. Alter your program so that it creates a Triangle object and passes it to draw_shape()
- Explain your observations

Copyright © James Raynard 2021

Function call arguments code

```
class Drawable {
        public:
        void draw() const { cout << "What am I supposed to do here?\n"; }
};

class Circle : public Drawable {
        public:
        void draw() const { cout << "I'm a circle!\n"; }
};

void draw_shape(const Drawable& dref) {
   dref.draw();
}
```

Copyright © James Raynard 2021

Polymorphism

- Explain briefly what is meant by "polymorphism"
- How do the programs in this video demonstrate polymorphism?

Copyright © James Raynard 2021

Pure Virtual Member Functions

- What is meant by a "pure virtual member function"?
- What effect does making a member function pure virtual have?
- In the program you wrote in the previous exercise, change Drawable::draw() to be a pure virtual function
- Compile and run your program again. What happens if you try to create a Drawable object?

Copyright © James Raynard 2021

Abstract Base Class

- What is an "abstract base class"?
- Why is it useful?

Copyright © James Raynard 2021

Virtual destructor in base class

- Explain why a base class should always have a virtual destructor implemented

Copyright © James Raynard 2021